Annual Report Oct 1, 1989-Sep 30 1990

1 Numerical Productivity Measures

K.T.Narayana
Department of Computer Science
Whitmore Laboratory
Pennsylvania State Univesrity
University Park, Pa16802
(Currently with Odyssey Research Associates)
Phone: 607-277-2020
Email: oravax!narayana@wrath.cs.cornell.edu

Grant Title: Real-Time System Specification and Verification
Grant Number: N00014-89-J-1171
Reporting Period: Oct,1 1989- Aug 15, 1990

Published Papers: 3
Papers Submitted for Publication: 2 (1 Accepted)
Technical Reports: 6
Graduate Student Supported: Eric Shade (half time from Jan 1, 1990-May 15,1990.)
Principal Investigator: Summer 1990 and 10

K.T.Narayana
Department of Computer Science
Whitmore Laboratory
Pennsylvania State Univesrity
University Park, Pa16802
(Currently with Odyssey Research Associates)
Phone: 607-277-2020
Email: oravax!narayana@wrath.cs.cornell.edu
Grant Title: Real-Time System Specification and Verification
Grant Number: N00014-89-J-1171
Reporting Period: Oct,1 1989- Aug 15, 1990

2 Summary of Technical Progress

A major revision was made to the formal semantic model for real-time concurrency under limited parallelism. The revision was required to simplify the model. The basic elements of the revision dealt with observing processes against the local clocks of processors and relating those observations to the global clock. The paper has been accepted for publication in Information and Computation subject to these revisions. This work has been done in collaboration with the graduate student Eric Shade.

Several notational simplifications have been broughtforth in the formal specification of the dialog system. The specification notation is made to conform to the latest Z-Language Standard. A part of the work dealing with the *Formal Specification of a Look Manager* has been published in the special issue on formal methods of the IEEE Transactions on Software Engineering, Sep 1990. Another part dealing with the *Invariant Propoerties in the Dialog System* has appeared in the Proceedings of the ACM Workshop on Formal Methods, May 1990. The specification of the design of the complete dialog system is currently going through a second iteration. Meanwhile we spent a lot of effort in implementing a syntax directed incremental specification environment for the Z-notation. The environment interfaces nicely with the X-Window system.Currently type inference systems are being integrated to the incremental editing environment. This work was done in collaboration with the graduate student Sanjeev Dharap.

Some conceptual simplifications are brought to bear on the language concept of tri-sections. These simplification will enhance the power of the notation while retaining the capability for creating regions of maximal parallelism in an otherwise limited parallel execution model. The formal semantics of the concepts will appear as part of Eric Shade's doctoral dissertation.

K.T.Narayana
Department of Computer Science
Whitmore Laboratory
Pennsylvania State Univesrity
University Park, Pa16802
(Currently with Odyssey Research Associates)
Phone: 607-277-2020
Email: oravax!narayana@wrath.cs.cornell.edu
Grant Title: Real-Time System Specification and Verification
Grant Number: N00014-89-J-1171
Reporting Period: Oct,1 1989- Aug 15, 1990

Published Papers:

K.T. Narayana and A.A. Aaby, Specification of Real-Time Systems in Real-Time Temporal Interval Logic, *IEEE Real-Time Systems Symposium*, December 1988.

K.T. Narayana and S. Dharap, Formal Specification of a Look Manager, *IEEE Transactions on Software Engineering*, Sep 1990.

K.T. Narayana and S. Dharap, Invariant Properties in a Dialog System, *Proceedings of the ACM Workshop on Formal Methods 1990*, May 1990. Also to appear in SigSoft Notices.

Eric Shade and K.T. Narayana, Real-Time Semantics for Shared Variable Concurrency, *accepted with revision for Information and Computation*

Papers Submitted for Publication

Tri-Sections: a Real-Time Programming Concept (with Eric Shade) (Acta Informatica).

Invariant Properties in a Dialog System (with S. Dharap) (Formal Aspects of Computing).

Technical Reports:

K.T. Narayana and S. Dharap, Formal Specification of a Look Manager.

K.T. Narayana and S. Dharap, Invariant Properties in a Dialog System.

Eric Shade and K.T. Narayana, Real-Time Semantics for Shared Variable.

A.A. Aaby and K.T. Narayana, Synthesis of Hardware Elements from Temporal Interval Logic Specifications.

K.T. Narayana, A Formal Model and a Specification of the Design of a Dialog System.

K.T.Narayana
Department of Computer Science
Whitmore Laboratory
Pennsylvania State Univesrity
University Park, Pa16802
(Currently with Odyssey Research Associates)
Phone: 607-277-2020
Email: oravax!narayana@wrath.cs.cornell.edu
Grant Title: Real-Time System Specification and Verification
Grant Number: N00014-89-J-1171
Reporting Period: Oct,1 1989- Aug 15, 1990

2.1 Semantic Models for Real-time Concurrency

In recent years the development and the use of real-time systems has increased dramatically. Real-time systems are used in applications where timing behavior is absolutely critical, such as patient monitoring systems, fly-by-wire avionic systems, and manufacturing control systems. Although more sophisticated communication schemes are available, the shared-variable paradigm is frequently employed in real-time systems. Practical limitations, both physical and economic, often restrict the number of physical processors which are available. A system may even be composed of dissimilar physical processors, none of which run at the same speed. For example, a system may consist of a single high-speed processor which is used solely for critical tasks, along with one or more slower processors on which the remainder of the work is distributed.

Clearly, these factors have a tremendous influence on the behavior of a real-time system. As such, there is a need for semantic models which take the execution environment of a system into account. In this paper, we provide a semantic model for real-time concurrency which accounts for

- shared variables (the CREW, EREW and bus-arbitration models),
- limited parallelism, in which the number of processes may exceed the number of physical processors, and
- asynchronous processors, where the conceptual speed of each physical processor depends on a local clock.

We provide a denotational semantics for a small, but realistic, real-time concurrent programming language L. The language uses shared variables for process cooperation, and supports atomic actions, process synchronization, and a delay command for real-time control. It contains alternative and repetitive commands based on guarded commands DIJKSTRA (1976). Features of the language appear in various forms in real-time languages such as CHILL BRANQUART, LOUIS AND WODON (1982), Ada ADA (1983), and occam OCCAM (1984). The primitives of the language are sufficiently low-level that a rich variety of "high-level" constructs can be implemented and analyzed using the semantic model.

Our work is closely related to KOYMANS *et. al.* (1988), which provides a linear-history semantics for a real-time variant of CSP HOARE (1978) under maximal parallelism. In that model, each process has its own processor, and all processors are synchronized via a conceptual global clock SALWICKI AND MÜLDNER (1981). Their work is an extension of FRANCEZ, LEHMAN AND PNUELI (1984) to real-time concurrency.

Other semantic models for real-time concurrency based on maximal parallelism have been explored in GERTH AND BOUCHER (1987), HUIZING, GERTH, AND DE ROEVER (1987), and SHYAMASUNDAR, NARAYANA, AND PITASSI (1987). For a slightly different approach, see for example SCCS MILNER (1983) and REED AND ROSCOE (1985).

Execution Models

A real-time program is the parallel composition of one or more sequential processes. We do not consider nested parallelism in this work. Programs are executed on a MIMD (multiple-instruction, multiple-data) ***shared-memory machine*** which consists of one or more physical processors running concurrently.

Each physical processor has its own local memory, and in addition there is a single shared memory which can be accessed by any processor. For simplicity we assume that each shared variable in a program occupies one location in the shared memory. There are three basic models of shared variable access:

- CREW (concurrent read, exclusive write). Any number of processes may read from a single variable simultaneously. If a variable is being written to, no other read or write of that variable can occur simultaneously.
- EREW (exclusive read, exclusive write). There can be at most one read or write access of a variable at a time.
- Bus-arbitration. Access to the shared-memory is controlled by a single bus which can only service one request at a time. No simultaneous accesses of any kind are permitted, even of different variables.

In all of the models, a processor can only access one variable at a time, although it could conceivably be ***waiting*** to access more than one variable. We assume that all memory accesses will be resolved within a finite period of time. δ denotes the maximum amount of global time that a process must wait before it is granted access to a variable. The function δ is a parameter of the model, and is obviously dependent on ρ.

A real-time execution model must specify three things: the number of available processors, the speed of the processors, and the allocation of processors to processes. Throughout this work we assume that there are ρ processors, numbered from 1 to ρ. There are two basic execution models, the maximal and limited parallelism models. In the maximal parallelism model, ρ is equal to the number of processes. Each process is allocated a processor which it uses until program termination. The processors all run at the same speed, and are synchronized via a conceptual global clock. During one "tick" of the clock, all of the processors simultaneously perform one machine instruction. We assume that time is discrete.

In the limited parallelism model, ρ may be less than the number of processes. Therefore process ***scheduling*** is required to make good use of the available processors. In order to make our semantic model as general as possible, we make minimal assumptions about the scheduler. As long as it does not violate the program semantics, the scheduler is free to move processes between processors at will. The one requirement is that there must be a delay of at least one global time unit before a process is switched from one processor to another. This is to prevent pointless process swapping. We call such a scheduler an *instantaneous* scheduler. Of course, real schedulers are far more complex than this, but it provides an excellent basis for a formal model which can easily be enhanced to account for more realistic systems.

In both models, *maximal throughput* is enforced. Given a choice between executing an action or idling (possibly waiting for some event to occur), a processor must always choose to perform an action as soon as possible. Further, processors may not go unused if there is a process waiting for execution. This *locally* maximizes processor utilization, but does not guarantee that the (global) execution time of the program will be minimal.

The limited parallelism model can be generalized by considering *asynchronous* processors. In addition to the global clock, each processor has its own *local* clock. Each "tick" of a local clock takes a discrete non-zero amount of global time, which is the *speed* of the processor. The function θ maps processors to their speeds. For example, if $\theta(4) = 3$ and $\theta(1) = 2$, then processor 4 runs 50% more slowly than processor 1.

It is important to note that limited parallelism and *interleaving* are very distinct models. Interleaving models represent the minimal assumptions necessary to ensure (qualitatively) correct execution of concurrent programs. They impose the least possible implementation restrictions. Limited parallelism, on the other hand, is a real-time model which enforces maximal throughput, and makes the processor resources explicit. A limited parallelism model has three parameters: ρ, the number of physical processors; δ, the maximum delay for shared memory access; and θ, the processor speeds. By choosing ρ to equal the number of processes and choosing $\theta(i)$ to be one for $1 \leq i \leq \rho$, maximal parallelism is just a special case of limited parallelism.

In this work, we consider three languages. A language L_A which consists of shared variables and an assignment statement, sequential composition, and parallel composition. The denotational semantics formulates the verification conditions with respect to memory access schemes. We subsequently introduce a simple language L_S with synchronization and atomic actions. Because synchronization between processes, and entry to atomic actions is recognizable against a global clock, observing processes can be tricky. We seek to observe processes at two levels. In the a priori semantics, we observe process events irrespective of the processors, or their speeds. We then impose process speeds on the observed a priori semantics after modifying observations with respect to synchronization and entry to atomic action elements. This leads to a simpler semantics. Further at every action of a process, the process can potentially wait in the scheduler. This fact is taken into account. While composing the processes, maximal throughput is achieved; this means that no processor idles unless prohibited by synchronization constraints. Finally, we consider a full complement of the language L consisting of L_A and L_S and an alternative and repetetive construct. This development shows how concepts can interact when put together particularly in a limited parallelism environment. We also prove that maximal parallelism is a special case of limited parallelism. Finally in the work, we show how to impose scheduling disciplines like priorities, and placed processes etc.

This work is being done in collaboration with the graduate student Eric Shade.

3 Formal Design of Dialog Systems

This year we have sought to extend and simplify the formal design of the dialog system. This simplification and revision led to the publication of components of the system as two papers. Our central interest in this aspect has been to obtain an understanding of the Z specification language and apply the same to large scale industrial size problems. Since the notations combine graphical text, we felt the need for developing a syntax directed specification environment for the Z notation. We built an incremental editing environment in which Z schemas can be treated as windows in X-Window model. The initial environment was set up against the notation given in the book by Ian Hayes. Most recently a new standard for the Z notation has been defined by people at Oxford. We are currently redesigning the system so that it can accept the latest Z language. In addition work is in progress towards integrating type inference systems in the incremental environment. This work is being done in collaboration with the graduate student S. Dharap from Penn State.

Bibliography

ADA (1983), "The Programming Lnaguage Ada Reference Manual," *Lecture Notes in Comput. Sci.* **155**, Springer-Verlag, Berlin.

APT, K.R., DE ROEVER, W.P., AND FRANCEZ, N. (1980), A proof system for communicating sequential processes, *Trans. on Programming Lang. and Systems* **2** (3), pp. 359–385.

BRANQUART, P., LOUIS, G., AND WODON, P. (1982), "An Analytical Description of CHILL, the CCITT High-Level Language VI," *Lecture Notes in Comput. Sci.* **128**, Springer-Verlag, Berlin.

DIJKSTRA, E.W. (1968), Cooperating sequential processes, *in* "Programming Languages," Academic Press, New York.

FRANCEZ, N., LEHMAN, D., AND PNEULI, A. (1984), A linear-history semantics for languages for distributed programming, *Theoret. Comput. Sci.* **32**, pp. 25–46.

GERTH, R., AND BOUCHER, A. (1987), A timed failures model for extended communicating processes, *Lecture Notes in Comput. Sci.* **267**, pp. 95–114, Springer-Verlag, Berlin.

HUIZING, C., GERTH, R., AND DE ROEVER, W.P. (1987), Full abstraction of a real-time denotational semantics for an occam-like language, *in Proc. 14th ACM Symp. on Principles of Programming Lang.*, pp. 223–238.

HOARE, C.A.R. (1978), Communicating sequential processes, *Comm. ACM* **21** (8), pp. 666–676.

KOYMANS, R., SHYAMASUNDAR, R.K., DE ROEVER, W.P., GERTH, R., AND ARUN-KUMAR, S. (1988), Compositional denotational semantics for real-time distributed computing, *Information and Control* **79** (3), pp. 210–256.

MILNER, R. (1983), Calculi for synchrony and asynchrony, *Theoret. Comput. Sci.* **25**, pp. 267–310.

MISRA, J. AND CHANDY, K.M. (1981), Proofs of networks of processes, *IEEE TSE* **7** (7), pp. 412–426.

OCCAM (1984), "The Occam Language Reference Manual," Prentice-Hall, Englewood Cliffs, NJ.

REED, G.M., AND ROSCOE, A.W. (1986), A timed model for communicating sequential processes, *Lecture Notes in Comput. Sci.* **226**, pp. 314–323, Springer-Verlag, Berlin.

SALWICKI, A., AND MÜLDNER, T. (1981), On the algorithmic properties of concurrent programs, *Lecture Notes in Comput. Sci.* **125**, pp. 169–197, Springer-Verlag, Berlin.

SHYAMASUNDAR, R.K., NARAYANA, K.T., AND PITASSI, T. (1987), A semantics for nondeterministic asynchronous broadcast networks, *Lecture Notes in Comput. Sci.* **267**, pp. 72–83, Springer-Verlag, Berlin.

SOUNDARARAJAN, N. (1984), Axiomatic semantics of communicating sequential processes, *ACM Trans. on Programming Lang. and Systems* **6** (4), pp. 647–662.

WIDOM, J., GRIES, D., AND SCHNEIDER, F.B. (1987), Completeness and incompleteness of trace-based network proof systems, *in Proc. 14th ACM Symp. on Principles of Programming Lang.*, pp. 27–38.

ZWIERS, J., DE ROEVER, W.P., AND VAN EMDE BOAS, P. (1985), Compositionality and concurrent networks: soundness and completeness of a proof system, *Lecture Notes in Comput. Sci.* **194**, pp. 509–519, Springer-Verlag, Berlin.